M 7

```
   CCCCCCCCCCCC     OOOOOOOOO     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
   CCCCCCCCCCCC     OOOOOOOOO     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
   CCCCCCCCCCCC     OOOOOOOOO     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
CCC              OOO         OOO  BBB         BBB  RRR         RRR        TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR         RRR        TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR         RRR        TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR         RRR        TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR         RRR        TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR         RRR        TTT        LLL
CCC              OOO         OOO  BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
CCC              OOO         OOO  BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
CCC              OOO         OOO  BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR   RRR              TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR   RRR              TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR   RRR              TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR      RRR           TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR      RRR           TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR      RRR           TTT        LLL
   CCCCCCCCCCCC     OOOOOOOOO     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
   CCCCCCCCCCCC     OOOOOOOOO     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
   CCCCCCCCCCCC     OOOOOOOOO     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

```
**FILE**ID**COBINTARI                                        G 2



  CCCCCCCC    OOOOOO    BBBBBBBB      IIIIII    NN      NN  TTTTTTTTTT    AAAAAA    RRRRRRRR      IIIIII
  CCCCCCCC    OOOOOO    BBBBBBBB      IIIIII    NN      NN  TTTTTTTTTT    AAAAAA    RRRRRRRR      IIIIII
CC          OO      OO  BB      BB      II      NN      NN      TT      AA      AA  RR      RR      II
CC          OO      OO  BB      BB      II      NN      NN      TT      AA      AA  RR      RR      II
CC          OO      OO  BB      BB      II      NNNN    NN      TT      AA      AA  RR      RR      II
CC          OO      OO  BB      BB      II      NNNN    NN      TT      AA      AA  RR      RR      II
CC          OO      OO  BBBBBBBB        II      NN  NN  NN      TT      AA      AA  RRRRRRRR        II
CC          OO      OO  BBBBBBBB        II      NN  NN  NN      TT      AA      AA  RRRRRRRR        II
CC          OO      OO  BB      BB      II      NN    NNNN      TT      AAAAAAAAAA  RR  RR          II
CC          OO      OO  BB      BB      II      NN    NNNN      TT      AAAAAAAAAA  RR  RR          II
CC          OO      OO  BB      BB      II      NN      NN      TT      AA      AA  RR    RR        II          ....
CC          OO      OO  BB      BB      II      NN      NN      TT      AA      AA  RR    RR        II          ....
  CCCCCCCC    OOOOOO    BBBBBBBB      IIIIII    NN      NN      TT      AA      AA  RR      RR    IIIIII        ....
  CCCCCCCC    OOOOOO    BBBBBBBB      IIIIII    NN      NN      TT      AA      AA  RR      RR    IIIIII        ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
Table of contents
```

```
0000        .TITLE COB$INTARI       COBOL intermediate arithmetic
0000        .IDENT /1-019/          ; File: COBINTARI.MAR Edit:SBL1019
0000    ;
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000    ;
0000    ;
0000    ;
0000    ;       HISTORY:
0000    ;
0000    ;       AUTHOR:
0000    ;               Marty Jack, 15-Apr-1979
0000    ;
0000    ;       MODIFIED BY:
0000    ;
0000    ;
```

```
0000          .SBTTL HISTORY          ; Detailed current edit history
0000
0000    ; Edit history for Version 1 of COBINTARI.MAR
0000    ;
0000    ; 1-001 - original, with input and output multiplexors and CMPI.
0000    ;         MLJ 15-Apr-1979
0000    ; 1-002 - include code for COB$ADDI.
0000    ; 1-003 - include code for COB$SUBI.
0000    ;         Wm P Storey,  07-Jun-1979
0000    ; 1-004 - include code for COB$DIVI.
0000    ; 1-005 - include code for COB$MULI.
0000    ; 1-006 - fixed post-normaliztion bug in COB$ADDI.
0000    ;         P D Gilbert,  21-Jun-1979
0000    ; 1-007 - update codes for data type (including COBOL Intermediate)
0000    ;         R. Reichert, 11-Sept-1979
0000    ; 1-008 - Code to return value from all routines.  MLJ 11-Sep-1979
0000    ; 1-009 - Re-write of COB$MULI due to MULP bugs with overflow.
0000    ;         PDG 11-Sep-1979
0000    ; 1-010 - Delete SIGNAL from DIVI.  MLJ 14-Sep-79
0000    ; 1-011 - Delete COBEXPI CODE -- now in separate module COBEXPI.MAR
0000    ;         RKR 19-Sept-79.
0000    ; 1-012 - Add missing .EXTRN COB$_INTDIVZER.  MLJ 05-Oct-79
0000    ; 1-013 - Replace ADDP4 #0,  with CMPP4 #0, now that ECO fixes micro-code
0000    ;         problem with CMPP4.   WPS 16-Oct-1979
0000    ; 1-014 - Change LIB$SIGNAL references to LIB$STOP.
0000    ;         Cosmetic changes.  RKR 21-OCT-79
0000    ; 1-015 - Add checks for out-of-range CIT in CONVERT and FINISH.
0000    ;         RKR 30-OCT-79
0000    ; 1-016 - Fix loss of least significant digit when borrow from MSD of 1.
0000    ;         WPS 6-Nov-1979
0000    ; 1-017 - Fix detection of exponent overflow and underflow generated by
0000    ;         the operation of COB$ADDI and COB$SUBI.  Correct addressing
0000    ;         problem.
0000    ;         Make a special case of detecting the generation of a fraction
0000    ;         of all zeroes by COB$ADDI and COB$SUBI.  In this case we
0000    ;         force an exponent of zero and bypass normalization of fraction.
0000    ;         RKR 23-APR-80
0000    ; 1-018 - Changed branch to 'FINISH' in routine COB$DIVI at label 21$: to
0000    ;         a RET instruction since 'FINISH' expects the input argument to
0000    ;         be in the proper format, where in this case the argument is in
0000    ;         error and therefore was never put in the format expected by
0000    ;         'FINISH'.  LB  15-APR-81
0000    ; 1-019 - Use general mode addressing.  SBL 30-Nov-1981
0000    ;
0000
```

```
              0000          .SBTTL DECLARATIONS
              0000
              0000          .DSABL  GBL
              0000
              0000     ;
              0000     ; INCLUDE FILES:
              0000     ;
              0000          $DSCDEF
              0000          $INTDEF
              0000
              0000     ;
              0000     ; EXTERNAL SYMBOLS:
              0000     ;
              0000          .EXTRN  COB$CVTWI_R8            ; Word to intermediate
              0000          .EXTRN  COB$CVTLI_R8            ; Longword to intermediate
              0000          .EXTRN  COB$CVTQI_R8            ; Quadword to intermediate
              0000          .EXTRN  COB$CVTFI_R7            ; Floating to intermediate
              0000          .EXTRN  COB$CVTDI_R7            ; Double to intermediate
              0000          .EXTRN  COB$CVTPI_R9            ; Packed to intermediate
              0000          .EXTRN  COB$CVTIW_R8            ; Intermediate to word
              0000          .EXTRN  COB$CVTIL_R8            ; Intermediate to longword
              0000          .EXTRN  COB$CVTIQ_R8            ; Intermediate to quadword
              0000          .EXTRN  COB$CVTIF_R7            ; Intermediate to floating
              0000          .EXTRN  COB$CVTID_R7            ; Intermediate to double
              0000          .EXTRN  COB$CVTIP_R9            ; Intermediate to packed
              0000          .EXTRN  COB$_INVARG             ; Invalid argument
              0000          .EXTRN  COB$_INTRESOPE
              0000          .EXTRN  COB$_INTDIVZER
              0000          .EXTRN  COB$_INTEXPUND          ; Intermediate underflow
              0000          .EXTRN  COB$_INTEXPOVE          ; Intermediate underflow
              0000          .EXTRN  LIB$STOP
              0000
              0000     ;
              0000     ; MACROS:
              0000     ;
              0000
              0000     ;
              0000     ; PSECT DECLARATIONS
              0000     ;
          00000000          .PSECT  _COB$CODE       PIC, SHR, LONG, EXE, NOWRT
              0000
              0000     ;
              0000     ; EQUATED SYMBOLS:
              0000     ;
00000002      0000     INT$P_I_FRACT=  2                ; Temporary until Packed supported in MDL
              0000                                      ; Fraction field offset
              0000
```

```
                    0000     134 ; OWN STORAGE:
                    0000     135 ;
                    0000     136 ;+
                    0000     137 ; The following is a packed zero.  Usage of this constant should be replaced
                    0000     138 ; by immediate operands when the assembler is corrected to allow them.
                    0000     139 ;-
                0C  0000     140 P0:      .PACKED 0
                1C  0001     141 P1:      .PACKED 1
```

```
                         0002          .SBTTL  CONVERT         Internal routine to convert to intermediate
                         0002
                         0002  ;+
                         0002  ;       Call by JSB
                         0002  ;       R0 points to descriptor (class = S or SD)
                         0002  ;       R1 points to output area (12 bytes)
                         0002  ;       Returns intermediate that has prefered sign in packed decimal mantissa.
                         0002  ;-
                         0002
                         0002  CONVERT:
   1F    00    02 A0  8F 0002          CASEB   DSC$B_DTYPE(R0),#0,#31  ; Go to proper conversion code
                  00EA'  0007  10$:    .WORD   BAD_DT-10$              ;  0 Z
                  00EA'  0009          .WORD   BAD_DT-10$              ;  1 V
                  00EA'  000B          .WORD   BAD_DT-10$              ;  2 BU
                  00EA'  000D          .WORD   BAD_DT-10$              ;  3 WU
                  00EA'  000F          .WORD   BAD_DT-10$              ;  4 LU
                  00EA'  0011          .WORD   BAD_DT-10$              ;  5 QU
                  00EA'  0013          .WORD   BAD_DT-10$              ;  6 B
                  0043'  0015          .WORD   20$-10$                 ;  7 W
                  005C'  0017          .WORD   30$-10$                 ;  8 L
                  0075'  0019          .WORD   40$-10$                 ;  9 Q
                  008E'  001B          .WORD   50$-10$                 ; 10 F
                  009B'  001D          .WORD   60$-10$                 ; 11 D
                  00EA'  001F          .WORD   BAD_DT-10$              ; 12 FC
                  00EA'  0021          .WORD   BAD_DT-10$              ; 13 DC
                  00EA'  0023          .WORD   BAD_DT-10$              ; 14 T
                  00EA'  0025          .WORD   BAD_DT-10$              ; 15 NU
                  00EA'  0027          .WORD   BAD_DT-10$              ; 16 NL
                  00EA'  0029          .WORD   BAD_DT-10$              ; 17 NLO
                  00EA'  002B          .WORD   BAD_DT-10$              ; 18 NR
                  00EA'  002D          .WORD   BAD_DT-10$              ; 19 NRO
                  00EA'  002F          .WORD   BAD_DT-10$              ; 20 NZ
                  00A8'  0031          .WORD   70$=10$                 ; 21 P
                  00EA'  0033          .WORD   BAD_DT-10$              ; 22 ZI
                  00EA'  0035          .WORD   BAD_DT-10$              ; 23 ZEM
                  00EA'  0037          .WORD   BAD_DT-10$              ; 24 DSC
                  00EA'  0039          .WORD   BAD_DT-10$              ; 25 OU
                  00EA'  003B          .WORD   BAD_DT-10$              ; 26 O
                  00EA'  003D          .WORD   BAD_DT-10$              ; 27 G
                  00EA'  003F          .WORD   BAD_DT-10$              ; 28 H
                  00EA'  0041          .WORD   BAD_DT-10$              ; 29 GC
                  00EA'  0043          .WORD   BAD_DT-10$              ; 30 HC
                  00C4'  0045          .WORD   80$=10$                 ; 31 COBOL intermediate data type
            00A7    31   0047          BRW     BAD_DT
                         004A
                         004A  ;+
                         004A  ; Source is W
                         004A  ;-
              56    D4   004A  20$:    CLRL    R6                      ; Assume class S
   09    03 A0      91   004C          CMPB    DSC$B_CLASS(R0),#DSC$K_CLASS_SD
              04    12   0050          BNEQ    21$                     ; Branch if not class SD
   56    08 A0      98   0052          CVTBL   DSC$B_SCALE(R0),R6      ; Get scale factor
   57    04 A0      D0   0056  21$:    MOVL    DSC$A_POINTER(R0),R7    ; Get source address
       58    51     D0   005A          MOVL    R1,R8                   ; Get destination address
  00000000'GF       17   005D          JMP     G^COB$CVTWI_R8          ; Go to conversion routine
                         0063
                         0063  ;+
                          0063  ; Source is L
                          0063  ;-
              56    D4    0063  30$:    CLRL    R6                          ; Assume class S
     09    03 A0    91    0065          CMPB    DSC$B_CLASS(R0),#DSC$K_CLASS_SD
              04    12    0069          BNEQ    31$                         ; Branch if not class SD
     56    08 A0    98    006B          CVTBL   DSC$B_SCALE(R0),R6          ; Get scale factor
     57    04 A0    D0    006F  31$:    MOVL    DSC$A_POINTER(R0),R7        ; Get source address
        58    51    D0    0073          MOVL    R1,R8                       ; Get destination address
    00000000'GF     17    0076          JMP     G^COB$CVTLI_R8              ; Go to conversion routine
                          007C
                          007C  ;+
                          007C  ; Source is Q
                          007C  ;-
              56    D4    007C  40$:    CLRL    R6                          ; Assume class S
     09    03 A0    91    007E          CMPB    DSC$B_CLASS(R0),#DSC$K_CLASS_SD
              04    12    0082          BNEQ    41$                         ; Branch if not class SD
     56    08 A0    98    0084          CVTBL   DSC$B_SCALE(R0),R6          ; Get scale factor
     57    04 A0    D0    0088  41$:    MOVL    DSC$A_POINTER(R0),R7        ; Get source address
        58    51    D0    008C          MOVL    R1,R8                       ; Get destination address
    00000000'GF     17    008F          JMP     G^COB$CVTQI_R8              ; Go to conversion routine
                          0095
                          0095  ;+
                          0095  ; Source is F
                          0095  ;-
     56    04 A0    D0    0095  50$:    MOVL    DSC$A_POINTER(R0),R6        ; Get source address
        57    51    D0    0099          MOVL    R1,R7                       ; Get destination address
    00000000'GF     17    009C          JMP     G^COB$CVTFI_R7              ; Go to conversion routine
                          00A2
                          00A2  ;+
                          00A2  ; Source is D
                          00A2  ;-
     56    04 A0    D0    00A2  60$:    MOVL    DSC$A_POINTER(R0),R6        ; Get source address
        57    51    D0    00A6          MOVL    R1,R7                       ; Get destination address
    00000000'GF     17    00A9          JMP     G^COB$CVTDI_R7              ; Go to conversion routine
                          00AF
                          00AF  ;+
                          00AF  ; Source is P
                          00AF  ;-
              56    D4    00AF  70$:    CLRL    R6                          ; Assume class S
     09    03 A0    91    00B1          CMPB    DSC$B_CLASS(R0),#DSC$K_CLASS_SD
              04    12    00B5          BNEQ    71$                         ; Branch if not class SD
     56    08 A0    98    00B7          CVTBL   DSC$B_SCALE(R0),R6          ; Get scale factor
        57    60    3C    00BB  71$:    MOVZWL  DSC$W_LENGTH(R0),R7         ; Get source length
     58    04 A0    D0    00BE          MOVL    DSC$A_POINTER(R0),R8        ; Get source address
        59    51    D0    00C2          MOVL    R1,R9                       ; Get destination address
    00000000'GF     17    00C5          JMP     G^COB$CVTPI_R9              ; Go to conversion routine
                          00CB
                          00CB  ;+
                          00CB  ; Source is intermediate
                          00CB  ;-
     50    04 A0    D0    00CB  80$:    MOVL    DSC$A_POINTER(R0),R0        ; Get source address
   0063 8F    60    B1    00CF          CMPW    INT$W_I_EXP(R0), #INT$K_I_EXP_HI ; Bigger than max ?
              0E    14    00D4          BGTR    81$                         ; Yes, overflow
   FF9D 8F    60    B1    00D6          CMPW    INT$W_I_EXP(R0), #INT$K_I_EXP_LO ; Less than min ?
              07    19    00DB          BLSS    81$                         ; Yes, underflow
        81    80    7D    00DD          MOVQ    (R0)+,(R1)+                 ; Copy 8 bytes
        61    60    D0    00E0          MOVL    (R0),(R1)                   ; Copy 4 more bytes
```

```
                          05  00E3    257           RSB                          ; Done
         00000000'8F      DD  00E4    258 81$:      PUSHL   #COB$_INTRESOPE      ; Intermediate reserved operand
 00000000'GF      01      FB  00EA    259           CALLS   #1,G^LIB$STOP        ; Signal the error
                              00F1    260
                              00F1    261 ;+
                              00F1    262 ; Here if not a supported data type.
                              00F1    263 ;-
         00000000'8F      DD  00F1    264 BAD_DT:   PUSHL   #COB$_INVARG         ; "Invalid argument list"
 00000000'GF      01      FB  00F7    265           CALLS   #1,G^LIB$STOP
```

```
                        00FE          .ENABL  LSB
                        00FE          .SBTTL  COB$SUBI        Subtract intermediate temporary
                        00FE
                        00FE  ;++
                        00FE  ; FUNCTIONAL DESCRIPTION:
                        00FE  ;
                        00FE  ;       Accept any two supported data types as input, convert them to
                        00FE  ;       Intermediate, subtract them, convert the Intermediate result to the
                        00FE  ;       data type of the output argument, and return.
                        00FE  ;
                        00FE  ; CALLING SEQUENCE:
                        00FE  ;
                        00FE  ;       COB$SUBI (SUBTRAHEND.rx.dx, MINUEND.rx.dx, DIFFERENCE.wx.dx)
                        00FE  ;
                        00FE  ; INPUT PARAMETERS:
                        00FE  ;
                        00FE  ;       SUBTRAHEND.rx.dx        The operand to the right of the operator
                        00FE  ;       MINUEND.rx.dx           The operand to the left of the operator
                        00FE  ;
                        00FE  ; IMPLICIT INPUTS:
                        00FE  ;
                        00FE  ;       NONE
                        00FE  ;
                        00FE  ; OUTPUT PARAMETERS:
                        00FE  ;
                        00FE  ;       DIFFERENCE.wx.dx        The difference of MINUEND - SUBTRAHEND
                        00FE  ;
                        00FE  ; IMPLICIT OUTPUTS:
                        00FE  ;
                        00FE  ;       NONE
                        00FE  ;
                        00FE  ; FUNCTION VALUE:
                        00FE  ;
                        00FE  ;       NONE
                        00FE  ;
                        00FE  ; SIDE EFFECTS:
                        00FE  ;
                        00FE  ;       NONE
                        00FE  ;--
                        00FE
                  03FC  00FE          .ENTRY  COB$SUBI,-
                        0100                  ^M<R2,R3,R4,R5,R6,R7,R8,R9>
                        0100
      5E     24     C2  0100          SUBL2   #<3*INT$K_I_LEN>,SP     ; Allocate space for 3 intermediates.
                        0103
  50      04 AC     D0  0103          MOVL    4(AP),R0                ; R0 now points to SUBTRAHEND.
  51      18 AE     9E  0107          MOVAB   <2*INT$K_I_LEN>(SP),R1  ; R1 now points to stack temp SUBTRAHEND.
            FEF4    30  010B          BSBW    CONVERT                 ; Convert operand1.
                        010E
  23 AE      01     8C  010E          XORB2   #1,             -       ; Change sign of SUBTRAHEND.
                        0112          <INT$K_I_FRACT_L-1>     -       ;
                        0112          +INT$P_I_FRACT          -       ;
                        0112          +<2*INT$K_I_LEN>(SP)            ;
             10     11  0112          BRB     10$                     ; Join COB$ADDI code.
```

```
                        0114            .SBTTL  COB$ADDI        Add intermediate temporary
                        0114
                        0114    ;++
                        0114    ; FUNCTIONAL DESCRIPTION:
                        0114    ;
                        0114    ;       Accept any two supported data types as input, convert them to
                        0114    ;       Intermediate, add them, convert the Intermediate result to the data
                        0114    ;       type of the output argument, and return.
                        0114    ;
                        0114    ; CALLING SEQUENCE:
                        0114    ;
                        0114    ;       COB$ADDI (ADDEND2.rx.dx, ADDEND1.rx.dx, SUM.wx.dx)
                        0114    ;
                        0114    ; INPUT PARAMETERS:
                        0114    ;
                        0114    ;       ADDEND2.rx.dx           The operand to the right of the operator
                        0114    ;       ADDEND1.rx.dx           The operand to the left of the operator
                        0114    ;
                        0114    ; IMPLICIT INPUTS:
                        0114    ;
                        0114    ;       INT$K_I_FRACT_D  must be even.
                        0114    ;
                        0114    ; OUTPUT PARAMETERS:
                        0114    ;
                        0114    ;       SUM.wx.dx               The sum of ADDEND1 + ADDEND2
                        0114    ;
                        0114    ; IMPLICIT OUTPUTS:
                        0114    ;
                        0114    ;       NONE
                        0114    ;
                        0114    ; FUNCTION VALUE:
                        0114    ;
                        0114    ;       NONE
                        0114    ;
                        0114    ; SIDE EFFECTS:
                        0114    ;
                        0114    ;       NONE
                        0114    ;--
                        0114
              00000000  0114            .IF     NE,<INT$K_I_FRACT_D -<2 * <INT$K_I_FRACT_D / 2>>>
                        0114            .ERROR          ; INT$K_I_FRACT_D must be even.
                        0114            .ENDC
                        0114
                  03FC  0114            .ENTRY  COB$ADDI,-
                        0116                    ^M<R2,R3,R4,R5,R6,R7,R8,R9>
      5E     24     C2  0116            SUBL2   #<3*INT$K_I_LEN>,SP     ; Allocate space for 3 intermediates.
                        0119
   50     04 AC     D0  0119            MOVL    4(AP),R0                ; R0 now points to ADDEND2.
   51     18 AE     9E  011D            MOVAB   <2*INT$K_I_LEN>(SP),R1  ; R1 now points to stack temp ADDEND2.
            FEDE    30  0121            BSBW    CONVERT                 ; Convert operand1.
                        0124    10$:                                    ; Subtract code joins here.
   50     08 AC     D0  0124            MOVL    8(AP),R0                ; R0 now points to ADDEND1.
   51     0C AE     9E  0128            MOVAB   INT$K_I_LEN(SP),R1      ; R1 now points to stack temp ADDEND1.
            FED3    30  012C            BSBW    CONVERT                 ; Convert operand2.
                        012F
                        012F
                        012F    ;
```

```
                    012F  ;     If the value of one intermediate is zero, the result of the add is the
                    012F  ;     other operand.
                    012F  ;
                    012F  ;     If the fraction contains a zero, the value of the intermediate temporary
                    012F  ;     datum is zero, regardless of the magnitude of the exponent.  It is only
                    012F  ;     convention (and hence can not be guaranteed) that if the fraction is 0
                    012F  ;     the exponent is 0.  Since the fraction part is normalized, the only
                    012F  ;     time that the low address byte of the fraction part is zero
                    012F  ;     is when the fraction part is zero.
                    012F  ;
                    012F  ;
     1A AE     95   012F          TSTB    <2*INT$K_I_LEN> -          ; Is ADDEND2 zero?
                    0132                  +INT$P_I_FRACT(SP)         ;
        05     12   0132          BNEQ    20$                        ; If NEQ, ADDEND2 is non-zero.
  5E    0C     C0   0134          ADDL2   #1*INT$K_I_LEN,SP          ; SP now points to other operand.
        08     11   0137          BRB     30$                        ; Join common code for word branch.
                    0139
                    0139  20$:                                       ;
     0E AE     95   0139          TSTB    <1*INT$K_I_LEN> -          ; Is ADDEND1 zero?
                    013C                  +INT$P_I_FRACT(SP)         ;
        09     12   013C          BNEQ    40$                        ; If NEQ, ADDEND1 is non-zero.
  5E    18     C0   013E          ADDL2   #2*INT$K_I_LEN,SP          ; SP now points to non-zero operand.
                    0141  30$:                                       ;
  50    01     D0   0141          MOVL    #1,R0                      ; Indicate success
      0255     31   0144          BRW     FINISH                     ; Convert (SP) to destination and return
                    0147
                    0147  40$:                                       ;
                    0147
                    0147  ;     As the fractional part of the intermediate temp is normalized, decimal
                    0147  ;     point alignment must be done before the actual add can be performed.
                    0147
                    0147  ;
                    0147  ;     Calculate difference between exponent of ADDEND1 and exponent of ADDEND2.
                    0147  ;
     18 AE     A3   0147          SUBW3   INT$W_I_EXP+<2*INT$K_I_LEN>(SP),-
     0C AE          014A                  INT$W_I_EXP+<1*INT$K_I_LEN>(SP),-
        56          014C                  R6                         ; R6 = e1 - e2
                    014D
        19     19   014D          BLSS    80$                        ; If LSS, e1 < e2
        0A     12   014F          BNEQ    70$                        ; If NEQ, e1 > e2
                    0151
                    0151  ;
                    0151  ;     At this point, exponents are equal.  According to Knuth Vol 2, p 218,
                    0151  ;     this has a frequency of occurrence of .47 for a radix of 10.
                    0151  ;
  57 18 AE     DE   0151          MOVAL   <2*INT$K_I_LEN>(SP),R7     ; R7 points to ADDEND2
  5E    0C     C0   0155          ADDL2   #INT$K_I_LEN,SP            ; SP points to ADDEND1
      0086     31   0158          BRW     120$                       ; Go do ADD.
                    015B
                    015B  ;
                    015B  ;     The SUBTRACT has established which number is larger;
                    015B  ;     that is,  which number has the larger exponent.
                    015B  ;     Set up R7 and R8 accordingly.
                    015B  ;
                    015B  70$:                                       ; e1 > e2
  56    56     AE   015B          MNEGW   R6,R6                      ; Make shift count negative.
  57 0C AE     9E   015E          MOVAB   <1*INT$K_I_LEN>(SP),R7     ; R7 is intermediate with larger exp.
```

```
                58    18 AE    9E  0162                 MOVAB   <2*INT$K_I_LEN>(SP),R8  ; R8 is intermediate with smaller exp.
                         08    11  0166                 BRB     90$                     ; Go do scaling.
                                   0168
                                   0168     80$:                                        ; e1 < e2
                57    18 AE    9E  0168                 MOVAB   <2*INT$K_I_LEN>(SP),R7  ; R7 is intermediate with larger exp.
                58    0C AE    9E  016C                 MOVAB   <1*INT$K_I_LEN>(SP),R8  ; R8 is intermediate with smaller exp.
                                   0170     90$:                                        ;
                                   0170
                                   0170     ;
                                   0170     ;   Ensure that the absolute value of the difference between exponents
                                   0170     ;   is less than or equal to INT$K_I_FRACT_D;
                                   0170     ;
          FFEE 8F     56       B1  0170                 CMPW    R6,#-<INT$K_I_FRACT_D>  ;
                      03       18  0175                 BGEQ    95$                     ; If GEQ, difference in range.
                56    13       CE  0177                 MNEGL   #INT$K_I_FRACT_D+1,R6   ; Set diff to max negative.
                                   017A     95$:                                        ;
                                   017A
                                   017A     ;
                                   017A     ;   Scale the number with the smaller exponent
                                   017A     ;   into the stack temporary for the sum.
                                   017A     ;
                6E    67       B0  017A                 MOVW    INT$W_I_EXP(R7),        -; Larger exponent becomes exponent
                                   017D                         INT$W_I_EXP(SP)         ;        of stack temp SUM.
                                   017D
  12    00    02 A8    12   56 F8  017D                 ASHP    R6,     -               ; Scale by the difference of exponents
                          02 AE    0184
                                   0186                         #INT$K_I_FRACT_D,       -;       the intermediate with
                                   0186                         INT$P_I_FRACT(R8),      -;       the smaller exponent
                                   0186                         #0,     -               ;       with no rounding
                                   0186                         #INT$K_I_FRACT_D,       -;       and with standard length
                                   0186                         INT$P_I_FRACT(SP)       ;       into the stack temp SUM.
                                   0186     ;
                                   0186     ;   Since this operation is taking place with infinite precision (only
                                   0186     ;   to be truncated to 18-digits), the digits that were just shifted off
                                   0186     ;   must be considered.  The effect of these digits is to contribute
                                   0186     ;   a one in the low order position only if
                                   0186     ;       a.)  the signs of the numbers being added are different
                                   0186     ;   and
                                   0186     ;       b.)  any of the digits just shifted out were non-zero.
                                   0186     ;
                                   0186
                      08 A8    8D  0186                 XORB3   <INT$K_I_FRACT_L-1>+INT$P_I_FRACT(R8),- ; Are signs different
                      08 A7        0189                         <INT$K_I_FRACT_L-1>+INT$P_I_FRACT(R7),- ;       or same?
                         59        018B                         R9
                      52 59    E9  018C                 BLBC    R9,120$                 ; If LBC, sign same.
                                   018F
                                   018F     ;
                                   018F     ;   As the signs are different, we have to be concerned about borrowing.
                                   018F     ;   Borrowing will only be a problem if the most significant digit of the
                                   018F     ;   number with the larger exponent is a 1 (this occurs 30% of the time).
                                   018F     ;   In that case, we need a guard digit to insure INT$K_I_FRACT_D digits
                                   018F     ;   of accuracy.  We will make use of the fact that INT$K_I_FRACT_D is even,
                                   018F     ;   and that therefore we have an extra digit available at INT$K_I_FRACT_D + 1.
                                   018F     ;   Note that we have to scale BOTH numbers.
                                   018F     ;
                                   018F
                                   018F
```

```
             02 A7    01    91  018F           CMPB    #^X01,INT$P_I_FRACT(R7) ; R7 is number with larger exponent.
                      18    12  0193           BNEQ    99$                     ; If NEQ, most significant digit not 1.
                                0195
                      01    F8  0195           ASHP    #1,-                    ; Effectively multiply by 10
                      12        0197                   #INT$K_I_FRACT_D,-      ; the appropriate number of digits
                   02 A7        0198                   INT$P_I_FRACT(R7),-     ; the number with the larger exponent
                      00        019A                   #0,-                    ; with no rounding
                      13        019B                   #INT$K_I_FRACT_D+1,-    ; with the extra digit
                   02 AE        019C                   INT$P_I_FRACT(SP)       ; into the stack result.
                                019E
                67    01    A3  019E           SUBW3   #1,INT$W_I_EXP(R7),-    ; Larger exponenet becomes exponent
                      6E        01A1                   INT$W_I_EXP(SP)         ;       of stack result.
                                01A2
                      56    B6  01A2           INCW    R6                      ; Shift smaller 1 less.
                      56    F8  01A4           ASHP    R6,-                    ; Shift down
                                01A6                   #INT$K_I_FRACT_D,-      ; the appropriate number of digits
             02 A8    12        01A6                   INT$P_I_FRACT(R8),-     ; the number with the smaller exponent
                      00        01A9                   #0,-                    ; with no rounding
                      13        01AA                   #INT$K_I_FRACT_D+1,-    ; with the extra digit
                   02 A7        01AB                   INT$P_I_FRACT(R7)       ; into the available other.
                                01AD
                                01AD
                                01AD  99$:
                                01AD  ;
                                01AD  ;   Non-zero digits that were shifted out contribute either
                                01AD  ;   +1 or -1 depending on the sign of the number.
                                01AD  ;
           59    56   02    A7  01AD           DIVW3   #2,R6,R9                        ; Convert from digits to bytes.
                 59   59    32  01B1           CVTWL   R9,R9                           ; Make number of bytes a longword.
                 55   56    AE  01B4           MNEGW   R6,R5                           ; R5 is number of digits shifted out.
                   06 56    E8  01B7           BLBS    R6,100$                         ; If LBS, number of digits odd.
        0B A849    F0 BF    8A  01BA           BICB    #^XF0,           -              ; Make high nibble zero.
                                01C0                   <INT$K_I_FRACT_L-1>+INT$P_I_FRACT(R8)[R9]
                                01C0  100$:                                            ;
0B A849    55    FE3B CF  01    37  01C0       CMPP4   #1,P0,R5,                     -; Were any of the digits shifted out
                                01C9                   <INT$K_I_FRACT_L-1>+INT$P_I_FRACT(R8)[R9] ;     non-zero?
                      16    13  01C9           BEQL    120$                            ; If EQL, all shifted out digits zero.
                                01CB
                                01CB  ;
                                01CB  ;   Increase its absolute value by one.
                                01CB  ;
             0A 0B A8       E9  01CB           BLBC    <INT$K_I_FRACT_L-1>-    ; If LBC, sign positive.
                                01CF                   +INT$P_I_FRACT(R8),110$ ;
       13    FE2D CF    01  22  01CF           SUBP4   #1,P1,#INT$K_I_FRACT_D+1,-; Contribute by negative 1.
                   02 AE        01D5                   INT$P_I_FRACT(SP)       ;
                      08    11  01D7           BRB     120$                    ; Join common code.
                                01D9                                           ;
                                01D9  110$:                                    ;
       13    FE23 CF    01  20  01D9           ADDP4   #1,P1,#INT$K_I_FRACT_D+1,-; Contribute by positive 1.
                   02 AE        01DF                   INT$P_I_FRACT(SP)       ;
                                01E1  120$:                                    ;
                                01E1
                                01E1  ;
                                01E1  ;   At this point, all scaling and adjustments have been made.
                                01E1  ;   The stack temp contains a number and approximate exponent.
                                01E1  ;   R7 points to the other number.
                                01E1  ;
```

```
                                 01E1   549 ;   Note that INT$K_I_FRACT_D+1 digit serves as a guard digit
                                 01E1   550 ;   for both carry and borrow.
                                 01E1   551 ;
     02 A7   13   02 AE   13  20 01E1   552         ADDP4   #INT$K_I_FRACT_D+1,     -; Finally, the actual add is done.
                                 01E8   553                 INT$P_I_FRACT(SP),      -;
                                 01E8   554                 #INT$K_I_FRACT_D+1,     -;
                                 01E8   555                 INT$P_I_FRACT(R7)        ;
                                 01E8   556
                          09  12 01E8   557                 BNEQ    129$             ; if nonzero, normalize
                          6E  B4 01EA   558                 CLRW    INT$W_I_EXP(SP)  ; set exponent to zero
                     12   00  F9 01EC   559                 CVTLP   #0, #INT$K_I_FRACT_D,-     ; set ans at (SP) to 0
                       02 AE     01EF   560                                 INT$P_I_FRACT(SP)  ;
                          1E  11 01F1   561                 BRB     131$             ; bypass normalization
                                 01F3   562
                                 01F3   563 ;
                                 01F3   564 ;   Post-normalization.
                                 01F3   565 ;   The most significant digit may be anywhere, due to a carry into the
                                 01F3   566 ;   nineteenth digit position, or a loss of significance (ex:12346-12345).
                                 01F3   567 ;   First we must find the first non-zero digit.
                                 01F3   568 ;
                                 01F3   569 129$:
                63   0A   00  3B 01F3   570         SKPC    #0,#INT$K_I_FRACT_L,(R3); Find first non-zero byte
                  61   F0 8F  93 01F7   571         BITB    #^XF0,(R1)                ; Is high digit of byte zero?
                          02  13 01FB   572         BEQL    130$                      ; Branch if so
                          52  D7 01FD   573         DECL    R2                        ; Otherwise, shift one less
                     51   53  C2 01FF   574 130$:   SUBL    R3,R1                     ; Compute byte offset of non-zero byte
                   50   6241  3E 0202   575         MOVAW   (R2)[R1],R0               ; Compute the shift amount
                     6E   50  A2 0206   576         SUBW    R0,INT$W_I_EXP(SP)        ; Twiddle the exponent
                                 0209   577
                     13   50  F8 0209   578         ASHP    R0,#INT$K_I_FRACT_D+1,-   ; Normalize the  fraction
                     00   63     020C   579                 (R3),#0,-                 ;
                          12     020E   580                 #INT$K_I_FRACT_D,-        ;
                       02 AE     020F   581                 INT$P_I_FRACT(SP)         ;
                                 0211   582 131$:
                     50   01  D0 0211   583         MOVL    #1,R0                     ; Indicate success
                        0185  31 0214   584         BRW     FINISH                    ; Convert to destination and return
                                 0217   585         .DSABL  LSB
                  0217            .SBTTL  COB$MULI        Multiply intermediate temporary
                  0217            .ENABL  LSB
                  0217      ;++
                  0217      ; FUNCTIONAL DESCRIPTION:
                  0217      ;
                  0217      ;       Accept any two supported data types as input, convert them to
                  0217      ;       Intermediate, multiply them, convert the Intermediate result to the
                  0217      ;       data type of the output argument, and return.
                  0217      ;
                  0217      ; CALLING SEQUENCE:
                  0217      ;
                  0217      ;       COB$MULI (MULTIPLIER.rx.dx, MULTIPLICAND.rx.dx, PRODUCT.wx.dx)
                  0217      ;
                  0217      ; INPUT PARAMETERS:
                  0217      ;
                  0217      ;       MULTIPLIER.rx.dx        The operand to the right of the operator
                  0217      ;       MULTIPLICAND.rx.dx      The operand to the left of the operator
                  0217      ;
                  0217      ; IMPLICIT INPUTS:
                  0217      ;
                  0217      ;       NONE
                  0217      ;
                  0217      ; OUTPUT PARAMETERS:
                  0217      ;
                  0217      ;       PRODUCT.wx.dx           The product MULTIPLICAND * MULTIPLIER
                  0217      ;
                  0217      ; IMPLICIT OUTPUTS:
                  0217      ;
                  0217      ;       NONE
                  0217      ;
                  0217      ; FUNCTION VALUE:
                  0217      ;
                  0217      ;       NONE
                  0217      ;
                  0217      ; SIDE EFFECTS:
                  0217      ;
                  0217      ;       NONE
                  0217      ;--
                  0217
                  0217      ; LOCAL SYMBOLS:        (To make this more readable)
                  0217      ;       (Note: we use the fact that INT$K_I_FRACT_D is even)
                  0217      ;
         0000000D 0217      D1      =       31-INT$K_I_FRACT_D      ; # of digs for first multiply
         00000006 0217      D2      =       INT$K_I_FRACT_D+T - D1  ; # of digs for second multiply
         00000003 0217      O1      =       D2/2                    ; Offset from fract of first multiply
                  0217      ;
                  0217      ;   Offsets from SP
                  0217      ;
         00000000 0217      MR      =       0                 ; Offset for M'plier & Product int temps
         0000000C 0217      MD      =       MR+INT$K_I_LEN    ; Offset for M'cand intermediate temp
         00000018 0217      Pr1     =       MD+INT$K_I_LEN    ; Offset for low product
         00000022 0217      Pr2     =       Pr1+<INT$K_I_FRACT_D/2+1>         ; Offset for high product
         0000002F 0217      SP_DECR =       Pr2+<<INT$K_I_FRACT_D+D2>/2+1>  ; Total to subtract from SP
                  0217
                  0217
             03FC 0217            .ENTRY  COB$MULI,-
                  0219                    ^M<R2,R3,R4,R5,R6,R7,R8,R9>
                  5E    2F   C2  0219    644          SUBL2   #SP_DECR,SP                 ; Two inter temps and a few extras
              50    04 AC   D0  021C    645          MOVL    4(AP),R0                    ; Convert operand 1
                  51    6E   9E  0220    646          MOVAB   MR(SP),R1
                      FDDC   30  0223    647          BSBW    CONVERT
              50    08 AC   D0  0226    648          MOVL    8(AP),R0                    ; Convert operand 2
              51    0C AE   9E  022A    649          MOVAB   MD(SP),R1
                      FDD1   30  022E    650          BSBW    CONVERT
                                 0231    651
                                 0231    652          MULP    -                           ; Calculate lower product
              11 AE    0D   25  0231    653                  #D1,01+INT$P_I_FRACT+MD(SP),-
              02 AE    12       0235    654                  #INT$K_I_FRACT_D,INT$P_I_FRACT+MR(SP),-
              18 AE    1F       0238    655                  #INT$K_I_FRACT_D+D1,Pr1(SP)
                 F0 8F   8B     023B    656          BICB3   #^XF0,-                     ; Put correct sign in middle of M'cand
                 17 AE          023E    657                  INT$K_I_FRACT_D/2+INT$P_I_FRACT+MD(SP),-
                    61          0240    658                  (R1)
                                 0241    659          MULP    -                           ; Calculate higher product (right sign)
              0E AE    06   25  0241    660                  #D2,INT$P_I_FRACT+MD(SP),-
              02 AE    12       0245    661                  #INT$K_I_FRACT_D,INT$P_I_FRACT+MR(SP),-
              22 AE    18       0248    662                  #INT$K_I_FRACT_D+D2,Pr2(SP)
                                 024B    663          MOVB    -                           ; Shorten lower product
                 0C A5     90   024B    664                  <INT$K_I_FRACT_D+D2>/2(R5),-
                 21 AE          024E    665                  <INT$K_I_FRACT_D/2>+Pr1(SP)
                                 0250    666          ADDP4   -                           ; Add the two products
              18 AE    13   20  0250    667                  #INT$K_I_FRACT_D+1,Pr1(SP),-
                 65    18       0254    668                  #INT$K_I_FRACT_D+D2,(R5)
                                 0256    669
                       50   D7  0256    670          DECL    R0                          ; Calculate amount to fiddle exponent
      50    50    65    00   EE  0258    671          EXTV    #0,(R5),R0,R0
                       50   D6  025D    672          INCL    R0                          ; Amount to fiddle exponent
           6E    0C AE   A0     025F    673          ADDW2   INT$W_I_EXP+MD(SP),INT$W_I_EXP+MR(SP)
                 6E    50   A2  0263    674          SUBW2   R0,INT$W_I_EXP+MR(SP)
                 50    06   82  0266    675          SUBB2   #D2,R0                      ; Calculate shift amount
                                 0269    676
           65    18    50   F8  0269    677          ASHP    R0,-                        ; Shift into result
                                 026D    678                  #INT$K_I_FRACT_D+D2,(R5),-
                       00       026D    679                  #0,-                        ; No rounding
              02 AE    12       026E    680                  #INT$K_I_FRACT_D,INT$P_I_FRACT+MR(SP)
                                 0271    681
                 50    01   D0  0271    682          MOVL    #1,R0                       ; Indicate success
                     0125   31  0274    683          BRW     FINISH                      ; Convert to destination and return
                                 0277    684
                                 0277    685          .DSABL  LSB
                 0277          .SBTTL  COB$DIVI        Divide intermediate temporary
                 0277
                 0277     ;++
                 0277     ; FUNCTIONAL DESCRIPTION:
                 0277     ;
                 0277     ;       Accept any two supported data types as input, convert them to
                 0277     ;       Intermediate, divide them, convert the Intermediate result to the data
                 0277     ;       type of the output argument, and return.
                 0277     ;
                 0277     ; CALLING SEQUENCE:
                 0277     ;
                 0277     ;       COB$DIVI        (DIVISOR.rx.dx, DIVIDEND.rx.dx, QUOTIENT.wx.dx)
                 0277     ;       COB$DIVI_OSE    (DIVISOR.rx.dx, DIVIDEND.rx.dx, QUOTIENT.wx.dx)
                 0277     ;
                 0277     ; INPUT PARAMETERS:
                 0277     ;
                 0277     ;       DIVISOR.rx.dx           The operand to the right of the operator
                 0277     ;       DIVIDEND.rx.dx          The operand to the left of the operator
                 0277     ;
                 0277     ; IMPLICIT INPUTS:
                 0277     ;
                 0277     ;       NONE
                 0277     ;
                 0277     ; OUTPUT PARAMETERS:
                 0277     ;
                 0277     ;       QUOTIENT.wx.dx          The quotient of DIVIDEND / DIVISOR
                 0277     ;
                 0277     ; IMPLICIT OUTPUTS:
                 0277     ;
                 0277     ;       If the entry is COB$DIVI, then signal COB$_INTDIVZER.
                 0277     ;
                 0277     ; FUNCTION VALUE:
                 0277     ;
                 0277     ;       NONE
                 0277     ;
                 0277     ; SIDE EFFECTS:
                 0277     ;
                 0277     ;       NONE
                 0277     ;--
                 0277
                 0277     ;
                 0277     ; EQUATED SYMBOLS:
                 0277     ;
        00000000 0277     t1      = 0                           ; Offset from SP
        00000015 0277     t2      = 21
        0000001C 0277     t3      = 28
        00000006 0277     t4      = t1+6
        0000002C 0277     dr      = 44                          ; Divisor
        00000038 0277     dd      = dr+INT$K_I_LEN              ; Dividend
        00000044 0277     ose     = dd+INT$K_I_LEN
        00000045 0277     sp_amt  = ose+1
                 0277
                 0277     ;
                 0277     ; Layout of temp storage as indexed from SP:
                 0277     ; (Divisor and Dividend temps are after these 44 bytes)
                 0277     ;
                 0277     ;01234567890123456789012345678901234567890123!
```

```
                        0277  ;t       t                 t       t           !
                        0277  ;1       4                 2       3           !
                        0277  ;dDDDDDDDDd00000s                              !  ASHP    dd,12,t1
                        0277  ;................          DDDDDDS             !  DIVP    dr,t1,t2
                        0277  ;................          .......s            !  BICB    t2+6
                        0277  ;................          .......,            !  MOVB    t2+6,R6
                        0277  ;................          .......,dDDDDDDDDDDDDDDs!  MULP    t2,dd,t3
                        0277  ;ooooooDDDDDDDDDs          ...... ................!  SUBP4   t3,t1
                        0277  ;       ..........,,ooos......                 !  ASHL    #24,t4+9,t4+11
                        0277  ;       .........oo..........                  !  CLRW    t4+9
                        0277  ;       ....................DDDDDS             !  DIVP    dr,t4,t2+6
                        0277  ;                          ...........S        !  BISB    R6,t2+11
                        0277  ;             ee           ...........         !  ...
                        0277  ;dDDDDDDDDS..              ...........         !  ASHP    t2,(SP)
                        0277  ;
                        0277  ; D =   2 digits in a byte
                        0277  ; d =   1 digit in byte (other digit is zero)
                        0277  ; S =   1 digit and a sign in byte
                        0277  ; s =   Sign in a byte (digit is zero)
                        0277  ; o =   Zero in byte
                        0277  ; . =   Useful information
                        0277  ; , =   Information used in this operation
                        0277  ;
                        0277
                  03FC  0277                 .ENTRY   COB$DIVI_OSE,-
                        0279                          ^M<R2,R3,R4,R5,R6,R7,R8,R9>
         50    01   98  0279                 CVTBL    #1,R0
               04   11  027C                 BRB      DIV_J
                  03FC  027E                 .ENTRY   COB$DIVI,-
                        0280                          ^M<R2,R3,R4,R5,R6,R7,R8,R9>
               50   D4  0280                 CLRL     R0
   5E   00000045 8F  C2  0282       DIV_J:   SUBL2    #sp_amt,SP                ; Get space for temp storage
       44 AE   50   90  0289                 MOVB     R0,ose(SP)                ; Remember which entry point
       50   04 AC   D0  028D                 MOVL     4(AP),R0                  ; Convert operand 1 (Divisor)
       51   2C AE   9E  0291                 MOVAB    dr(SP),R1                 ;
             FD6A   30  0295                 BSBW     CONVERT                   ;
            2E AE   95  0298                 TSTB     dr+INT$P_I_FRACT(SP)      ; Is divisor equal to zero?
               6C   13  029B                 BEQL     20$
       50   08 AC   D0  029D                 MOVL     8(AP),R0                  ; Convert operand 2 (Dividend)
       51   38 AE   9E  02A1                 MOVAB    dd(SP),R1                 ;
             FD5A   30  02A5                 BSBW     CONVERT                   ;
                        02A8
        12     0C   F8  02A8                 ASHP     #12,#INT$K_I_FRACT_D,-    ; Multiply Dividend by 10**12
            3A AE       02AB                          dd+INT$P_I_FRACT(SP),-    ; (dd)
               00       02AD                          #0,-
               1E       02AE                          #<12+INT$K_I_FRACT_D>,-
               6E       02AF                          (SP)                      ; (t1)
               12   27  02B0                 DIVP     #INT$K_I_FRACT_D,-        ; And divide by Divisor
            2E AE       02B2                          dr+INT$P_I_FRACT(SP),-    ; (dr)
               1E       02B4                          #<12+INT$K_I_FRACT_D>,-
               6E       02B5                          (SP),-                    ; (t1)
    15 AE      0D       02B6                          #<12+1>,t2(SP)            ; (t2)
 1B AE    F0 8F     8A  02B9                 BICB     #^XF0,<<12+1>/2+t2>(SP)   ; Zap the least significant digit(!!!)
    56     1B AE    90  02BE                 MOVB     <<12+1>/2+t2>(SP),R6      ; Save the true sign
               0D   25  02C2                 MULP     #<12+1>,-                 ; Multiply back (by Divisor)
               65       02C4                          (R5),-                    ; (t2)
               12       02C5                          #INT$K_I_FRACT_D,-
```

```
                  61        02C6   801                   (R1),-                     ; (dr)
                  1E        02C7   802                   #<12+INT$K_I_FRACT_D>,-
               1C AE        02C8   803                   t3(SP)                     ; (t3)
                  1E   22   02CA   804          SUBP4    #<12+INT$K_I_FRACT_D>,-    ; And subtract from Dividend*10**12...
                  65        02CC   805                   (R5),-                     ; (t3)
                  1E        02CD   806                   #<12+INT$K_I_FRACT_D>,-    ;            ... giving a 'remainder'
                  6E        02CE   807                   (SP)                       ; (t1)
                            02CF   808                                              ; Low INT$K_I_FRACT_D+1 digits are t4
                  18   78   02CF   809          ASHL     #<3*8>,-                   ; Multiply t4 by 10**10
               0F AE        02D1   810                   <<INT$K_I_FRACT_D+1>/2+t4>(SP),-
               11 AE        02D3   811                   <<INT$K_I_FRACT_D+1+10>/2+t4-3>(SP)
                            02D5   812                                              ;            (by moving the sign right...)
                            02D5   813                                              ;            (...and zapping the old sign)
               0F AE   B4   02D5   814          CLRW     <<INT$K_I_FRACT_D+1>/2+t4>(SP)
                  12   27   02D8   815          DIVP     #INT$K_I_FRACT_D,-         ; Divide 'remainder' by Divisor
               2E AE        02DA   816                   dr+INT$P_I_FRACT(SP),-     ; (dr)
                  1D        02DC   817                   #<INT$K_I_FRACT_D+1+10>,-
               06 AE        02DD   818                   t4(SP),-                   ; (t4)
                  0B        02DF   819                   #<1+10>,-
               1B AE        02E0   820                   <<12+1>/2+t2>(SP)          ; Putting it at low end of first DIVP
         20 AE    56   88   02E2   821          BISB     R6,<<12+1+10>/2+t2>(SP)    ; Put back true sign (if the 2nd DIVP
                            02E6   822                                              ;            gave 0, the sign may be wrong)
                            02E6   823 ;
                            02E6   824 ;     Temp-2 is now a 23-digit (12+1+10) packed item equal to:
                            02E6   825 ;        Z + [ ( Dividend x 10**12 - Divisor x Z ) / Divisor ],
                            02E6   826 ;      where [...] indicates integer truncation, and
                            02E6   827 ;        Z = ( [ [ Dividend x 10**12 / Divisor ] / 10 ] * 10 ) * 10**10
                            02E6   828 ;
            50    04   8E   02E6   829          MNEGB    #4,R0                      ; Shift amount (19-23)
   6E    38 AE 2C AE   A3   02E9   830          SUBW3    INT$W_I_EXP+dr(SP), -      ; Calculate exponent
                            02EF   831                   INT$W_I_EXP+dd(SP), -
                            02EF   832                   INT$W_I_EXP(SP)
         15 AE F0 8F   93   02EF   833          BITB     #^XF0,t2(SP)               ; Re-normalization needed?
                  04   13   02F4   834          BEQL     10$                        ; No
                  6E   B6   02F6   835          INCW     INT$W_I_EXP(SP)            ; Yes. Increase the exponent
                  50   97   02F8   836          DECB     R0                         ; Move right a little more
      15 AE    17 50   F8   02FA   837 10$:     ASHP     R0,#<12+1+10>,t2(SP),-     ; Shift into Quotient
               12 00        02FF   838                   #0,#INT$K_I_FRACT_D,-
               02 AE        0301   839                   INT$P_I_FRACT(SP)
               50 01   D0   0303   840          MOVL     #1,R0                      ; Indicate success
                0093   31   0306   841          BRW      FINISH                     ; Gee, that was easy
                            0309   842
                            0309   843 ;
                            0309   844 ;        The Divisor is zero
                            0309   845 ;
            0D 44 AE   E8   0309   846 20$:     BLBS     ose(SP),21$                ; Branch if entry is COB$DIVI_OSE
           00000000'8F DD   030D   847          PUSHL    #COB$_INTDIVZER
        00000000'GF 01 FB   0313   848          CALLS    #1,G^LIB$STOP
         5E    38 AE   9E   031A   849 21$:     MOVAB    dd(SP),SP                  ; Return dividend
                  50   D4   031E   850          CLRL     R0                         ; Indicate failure
                  04   0320        851          RET
```

```
                                 0321          .SBTTL  COB$CMPI        Compare intermediate temporary
                                 0321
                                 0321   ;++
                                 0321   ; FUNCTIONAL DESCRIPTION:
                                 0321   ;
                                 0321   ;       Accept any two supported data types as input, convert them to
                                 0321   ;       Intermediate, compare them, and return the result of comparison
                                 0321   ;       as value.
                                 0321   ;
                                 0321   ; CALLING SEQUENCE:
                                 0321   ;
                                 0321   ;       VALUE.wl.v = COB$CMPI (SRC1.rx.dx, SRC2.rx.dx)
                                 0321   ;
                                 0321   ; INPUT PARAMETERS:
                                 0321   ;
                                 0321   ;       SRC1.rx.dx              The operand to the left of the operator
                                 0321   ;       SRC2.rx.dx              The operand to the right of the operator
                                 0321   ;
                                 0321   ; IMPLICIT INPUTS:
                                 0321   ;
                                 0321   ;       NONE
                                 0321   ;
                                 0321   ; OUTPUT PARAMETERS:
                                 0321   ;
                                 0321   ;       NONE
                                 0321   ;
                                 0321   ; IMPLICIT OUTPUTS:
                                 0321   ;
                                 0321   ;       NONE
                                 0321   ;
                                 0321   ; FUNCTION VALUE:
                                 0321   ;
                                 0321   ;       VALUE.wl.v              -1 if SRC1 LSS SRC2
                                 0321   ;                                0 if SRC1 EQL SRC2
                                 0321   ;                               +1 if SRC1 GTR SRC2
                                 0321   ;
                                 0321   ; SIDE EFFECTS:
                                 0321   ;
                                 0321   ;       NONE
                                 0321   ;--
                                 0321
                           03FC  0321          .ENTRY  COB$CMPI,-
                                 0323                  ^M<R2,R3,R4,R5,R6,R7,R8,R9>
               5E     18     C2  0323          SUBL2   #<2*INT$K_I_LEN>,SP     ; Space for 2 intermediate temps
           50     04 AC      D0  0326          MOVL    4(AP),R0                ; Convert operand 1
           51     0C AE      9E  032A          MOVAB   INT$K_I_LEN(SP),R1      ;
                    FCD1     30  032E          BSBW    CONVERT                 ;
           50     08 AC      D0  0331          MOVL    8(AP),R0                ; Convert operand 2
               51     5E     D0  0335          MOVL    SP,R1                   ;
                    FCC7     30  0338          BSBW    CONVERT                 ;
                                 033B   ;
                                 033B   ; Case on the sign of the left operand.
                                 033B   ;
FCBD CF    01    0E AE   12  37  033B          CMPP4   #INT$K_I_FRACT_D,INT$P_I_FRACT+INT$K_I_LEN(SP),#1,P0
                         11  14  0343          BGTR    10$                     ; Br if left GTR 0
                         2E  19  0345          BLSS    20$                     ; Br if left LSS 0
                                 0347   ;
```

```
                                    0347  ; Here if the left operand is zero.  Case on the sign of the right operand.
                                    0347  ;
   FCB1 CF    01    02 AE     12    37  0347          CMPP4   #INT$K_I_FRACT_D,INT$P_I_FRACT(SP),#1,P0
                              43    14  034F          BGTR    30$                             ; Br if left EQL 0 and right GTR 0
                              45    19  0351          BLSS    40$                             ; Br if left EQL 0 and right LSS 0
                              50    D4  0353          CLRL    R0                              ; Set "left EQL right"
                                    04  0355          RET                                     ; Return
                                        0356  ;
                                        0356  ; Here if the left operand is positive.  If the right operand is nonpositive,
                                        0356  ; it must be smaller.  Otherwise, compare the exponents and then the fractions
                                        0356  ; if the exponents are equal.  Since both numbers are positive, the larger
                                        0356  ; magnitudes correspond to larger numbers.
                                        0356  ;
   FCA2 CF    01    02 AE     12    37  0356  10$:    CMPP4   #INT$K_I_FRACT_D,INT$P_I_FRACT(SP),#1,P0
                              38    15  035E          BLEQ    40$                             ; Br if left GTR 0 and right LEQ 0
                    6E    0C  AE    B1  0360          CMPW    INT$W_I_EXP+INT$K_I_LEN(SP),INT$W_I_EXP(SP)
                              32    14  0364          BGTR    40$                             ; Br if left exp GTR right exp
                              2C    19  0366          BLSS    30$                             ; Br if left exp LSS right exp
         02 AE     0E AE      12    35  0368          CMPP3   #INT$K_I_FRACT_D,INT$P_I_FRACT+INT$K_I_LEN(SP),INT$P_I_FRACT(SP)
                              28    14  036E          BGTR    40$                             ; Br if left frac GTR right frac
                              22    19  0370          BLSS    30$                             ; Br if left frac LSS right frac
                              50    D4  0372          CLRL    R0                              ; Set "left EQL right"
                                    04  0374          RET
                                        0375  ;
                                        0375  ; Here if the left operand is negative.  If the right operand is nonnegative,
                                        0375  ; it must be larger.  Otherwise, compare the exponents and then the fractions
                                        0375  ; if the exponents are equal.  Since both numbers are negative, the larger
                                        0375  ; magnitudes correspond to smaller numbers.
                                        0375  ;
   FC83 CF    01    02 AE     12    37  0375  20$:    CMPP4   #INT$K_I_FRACT_D,INT$P_I_FRACT(SP),#1,P0
                              15    18  037D          BGEQ    30$                             ; Br if left LSS 0 and right GEQ 0
                    6E    0C  AE    B1  037F          CMPW    INT$W_I_EXP+INT$K_I_LEN(SP),INT$W_I_EXP(SP)
                              13    19  0383          BLSS    40$                             ; Br if left exp LSS right exp
                              0D    14  0385          BGTR    30$                             ; Br if left exp GTR right exp
         02 AE     0E AE      12    35  0387          CMPP3   #INT$K_I_FRACT_D,INT$P_I_FRACT+INT$K_I_LEN(SP),INT$P_I_FRACT(SP)
                              09    14  038D          BGTR    40$                             ; Br if left frac GTR right frac
                              03    19  038F          BLSS    30$                             ; Br if left frac LSS right frac
                              50    D4  0391          CLRL    R0                              ; Set "left EQL right"
                                    04  0393          RET                                     ; Return
                                        0394  ;
                                        0394  ; Here to return +1 and -1 values.
                                        0394  ;
                        50    01    CE  0394  30$:    MNEGL   #1,R0                           ; Set "left LSS right"
                                    04  0397          RET                                     ; Return
                        50    01    D0  0398  40$:    MOVL    #1,R0                           ; Set "left GTR right"
                                    04  039B          RET                                     ; Return
```

```
                               039C    957          .SBTTL  FINISH          Convert to destination type and return
                               039C    958
                               039C    959 ;+
                               039C    960 ;        Enter by branch with (SP) containing the intermediate result
                               039C    961 ;        and 12(AP) pointing to the descriptor for the destination.
                               039C    962 ;        R0 contains routine status.
                               039C    963 ;-
                               039C    964
                               039C    965 FINISH:
              02 AE      95    039C    966          TSTB    INT$P_I_FRACT(SP)       ; is fraction zero ?
                 04      12    039F    967          BNEQ    8$                      ; no
                 6E      B4    03A1    968          CLRW    INT$W_I_EXP(SP)         ; force exponent to zero
                 0E      11    03A3    969          BRB     9$                      ; bypass overflow and underflow
                               03A5    970                                          ; checks
                               03A5    971 ;+
                               03A5    972 ; Check for out-of-range conditions first
                               03A5    973 ; We do the check here for all destination type so that we can report
                               03A5    974 ; overflow and underflow distinctly.  If we allow the flow to go
                               03A5    975 ; directly to various COB$CVTI_x routines, what will be reported
                               03A5    976 ; is COB$_INTRESOPE (which is not correct -- we just created the
                               03A5    977 ; exception and did not access it -- creating an exception should
                               03A5    978 ; distinguish between over_ and under_flow)
                               03A5    979 ;-
                               03A5    980
                               03A5    981 8$:
       0063 8F      6E      B1    03A5    982          CMPW    INT$W_I_EXP(SP), #INT$K_I_EXP_HI ; Bigger than max ?
                    55      14    03AA    983          BGTR    3$                               ; Yes, overflow
       FF9D 8F      6E      B1    03AC    984          CMPW    INT$W_I_EXP(SP), #INT$K_I_EXP_LO ; Less than min ?
                    56      19    03B1    985          BLSS    5$                               ; Yes, underflow
                                  03B3    986 9$:
                    50      DD    03B3    987          PUSHL   R0                      ; Save success status
                                  03B5    988                                          ; Result now at 4(SP)
                                  03B5    989
          50     0C AC      D0    03B5    990          MOVL    12(AP),R0               ; pick up the descriptor addr.
   1F     00     02 A0      8F    03B9    991          CASEB   DSC$B_DTYPE(R0),#0,#31
                          FD33    03BE    992 10$:     .WORD   BAD_DT-10$      ;  0 Z
                          FD33    03C0    993          .WORD   BAD_DT-10$      ;  1 V
                          FD33    03C2    994          .WORD   BAD_DT-10$      ;  2 BU
                          FD33    03C4    995          .WORD   BAD_DT-10$      ;  3 WU
                          FD33    03C6    996          .WORD   BAD_DT-10$      ;  4 LU
                          FD33    03C8    997          .WORD   BAD_DT-10$      ;  5 QU
                          FD33    03CA    998          .WORD   BAD_DT-10$      ;  6 B
                         0058'    03CC    999          .WORD   20$-10$         ;  7 W
                         0079'    03CE   1000          .WORD   30$-10$         ;  8 L
                         009A'    03D0   1001          .WORD   40$-10$         ;  9 Q
                         00BB'    03D2   1002          .WORD   50$-10$         ; 10 F
                         00CD'    03D4   1003          .WORD   60$-10$         ; 11 D
                          FD33    03D6   1004          .WORD   BAD_DT-10$      ; 12 FC
                          FD33    03D8   1005          .WORD   BAD_DT-10$      ; 13 DC
                          FD33    03DA   1006          .WORD   BAD_DT-10$      ; 14 T
                          FD33    03DC   1007          .WORD   BAD_DT-10$      ; 15 NU
                          FD33    03DE   1008          .WORD   BAD_DT-10$      ; 16 NL
                          FD33    03E0   1009          .WORD   BAD_DT-10$      ; 17 NLO
                          FD33    03E2   1010          .WORD   BAD_DT-10$      ; 18 NR
                          FD33    03E4   1011          .WORD   BAD_DT-10$      ; 19 NRO
                          FD33    03E6   1012          .WORD   BAD_DT-10$      ; 20 NZ
                         00DF'    03E8   1013          .WORD   70$-10$         ; 21 P
```

```
                  FD33  03EA          .WORD   BAD_DT-10$              ; 22 ZI
                  FD33  03EC          .WORD   BAD_DT-10$              ; 23 ZEM
                  FD33  03EE          .WORD   BAD_DT-10$              ; 24 DSC
                  FD33  03F0          .WORD   BAD_DT-10$              ; 25 OU
                  FD33  03F2          .WORD   BAD_DT-10$              ; 26 O
                  FD33  03F4          .WORD   BAD_DT-10$              ; 27 G
                  FD33  03F6          .WORD   BAD_DT-10$              ; 28 H
                  FD33  03F8          .WORD   BAD_DT-10$              ; 29 GC
                  FD33  03FA          .WORD   BAD_DT-10$              ; 30 HC
                 0103'  03FC          .WORD   80$-10$                 ; 31 COBOL intermediate data type
             FCF0   31  03FE          BRW     BAD_DT
                        0401
                        0401
                        0401  ;+
                        0401  ; CIT overflowed.
                        0401  ;-
                        0401  3$:
      00000000'8F   DD  0401          PUSHL   #COB$_INTEXPOVE         ; Overflow signal
               06   11  0407          BRB     6$                      ; go signal
                        0409
                        0409  ;+
                        0409  ; CIT underflow
                        0409  ;-
                        0409  5$:
      00000000'8F   DD  0409          PUSHL   #COB$_INTEXPUND         ; Underflow signal
 00000000'GF   01   FB  040F  6$:     CALLS   #1,G^LIB$STOP           ; Signal and stop.
                        0416
                        0416  ;+
                        0416  ; Destination is W
                        0416  ;-
               56   D4  0416  20$:    CLRL    R6                      ; Assume class S
       09   03 A0   91  0418          CMPB    DSC$B_CLASS(R0),#DSC$K_CLASS_SD
               07   12  041C          BNEQ    21$                     ; Branch if not class SD
       56   08 A0   98  041E          CVTBL   DSC$B_SCALE(R0),R6      ; Get scale factor
          56   56   CE  0422          MNEGL   R6,R6                   ; Negate scale factor
       57   04 AE   9E  0425  21$:    MOVAB   4(SP),R7                ; Get source address
       58   04 A0   D0  0429          MOVL    DSC$A_POINTER(R0),R8    ; Get destination address
      00000000'GF   16  042D          JSB     G^COB$CVTIW_R8          ; Go to conversion routine
          50   8E   D0  0433          MOVL    (SP)+,R0                ; Restore status
                    04  0436          RET                             ; Return
                        0437
                        0437  ;+
                        0437  ; Destination is L
                        0437  ;-
               56   D4  0437  30$:    CLRL    R6                      ; Assume class S
       09   03 A0   91  0439          CMPB    DSC$B_CLASS(R0),#DSC$K_CLASS_SD
               07   12  043D          BNEQ    31$                     ; Branch if not class SD
       56   08 A0   98  043F          CVTBL   DSC$B_SCALE(R0),R6      ; Get scale factor
          56   56   CE  0443          MNEGL   R6,R6                   ; Negate scale factor
       57   04 AE   9E  0446  31$:    MOVAB   4(SP),R7                ; Get source address
       58   04 A0   D0  044A          MOVL    DSC$A_POINTER(R0),R8    ; Get destination address
      00000000'GF   16  044E          JSB     G^COB$CVTIL_R8          ; Go to conversion routine
          50   8E   D0  0454          MOVL    (SP)+,R0                ; Restore status
                    04  0457          RET                             ; Return
                        0458
                        0458  ;+
                        0458  ; Destination is Q
```

```
                            0458   1071 ;-
                    56  D4  0458   1072 40$:    CLRL    R6                              ; Assume class S
           09    03 A0  91  045A   1073         CMPB    DSC$B_CLASS(R0),#DSC$K_CLASS_SD
                    07  12  045E   1074         BNEQ    41$                             ; Branch if not class SD
           56    08 A0  98  0460   1075         CVTBL   DSC$B_SCALE(R0),R6              ; Get negative of scale factor
                 56  56  CE  0464  1076         MNEGL   R6,R6                           ;
           57    04 AE  9E  0467   1077 41$:    MOVAB   4(SP),R7                        ; Get source address
           58    04 A0  D0  046B   1078         MOVL    DSC$A_POINTER(R0),R8            ; Get destination address
          00000000'GF   16  046F   1079         JSB     G^COB$CVTIQ_R8                  ; Go to conversion routine
                 50  8E  D0  0475  1080         MOVL    (SP)+,R0                        ; Restore status
                        04  0478   1081         RET                                     ; Return
                            0479   1082
                            0479   1083 ;+
                            0479   1084 ; Destination is F
                            0479   1085 ;-
           56    04 AE  9E  0479   1086 50$:    MOVAB   4(SP),R6                        ; Get source address
           57    04 A0  D0  047D   1087         MOVL    DSC$A_POINTER(R0),R7            ; Get destination address
          00000000'GF   16  0481   1088         JSB     G^COB$CVTIF_R7                  ; Go to conversion routine
                 50  8E  D0  0487  1089         MOVL    (SP)+,R0                        ; Restore status
                        04  048A   1090         RET                                     ; Return
                            048B   1091
                            048B   1092 ;+
                            048B   1093 ; Destination is D
                            048B   1094 ;-
           56    04 AE  9E  048B   1095 60$:    MOVAB   4(SP),R6                        ; Get source address
           57    04 A0  D0  048F   1096         MOVL    DSC$A_POINTER(R0),R7            ; Get destination address
          00000000'GF   16  0493   1097         JSB     G^COB$CVTID_R7                  ; Go to conversion routine
                 50  8E  D0  0499  1098         MOVL    (SP)+,R0                        ; Restore status
                        04  049C   1099         RET                                     ; Return
                            049D   1100
                            049D   1101 ;+
                            049D   1102 ; Destination is P
                            049D   1103 ;-
                    56  D4  049D   1104 70$:    CLRL    R6                              ; Assume class S
           09    03 A0  91  049F   1105         CMPB    DSC$B_CLASS(R0),#DSC$K_CLASS_SD
                    07  12  04A3   1106         BNEQ    71$                             ; Branch if not class SD
           56    08 A0  98  04A5   1107         CVTBL   DSC$B_SCALE(R0),R6              ; Get negative of scale factor
                 56  56  CE  04A9  1108         MNEGL   R6,R6                           ;
           57    04 AE  9E  04AC   1109 71$:    MOVAB   4(SP),R7                        ; Get source address
                 58  60  3C  04B0  1110         MOVZWL  DSC$W_LENGTH(R0),R8             ; Get destination length
           59    04 A0  D0  04B3   1111         MOVL    DSC$A_POINTER(R0),R9            ; Get destination address
          00000000'GF   16  04B7   1112         JSB     G^COB$CVTIP_R9                  ; Go to conversion routine
                 50  8E  D0  04BD  1113         MOVL    (SP)+,R0                        ; Restore status
                        04  04C0   1114         RET                                     ; Return
                            04C1   1115
                            04C1   1116 ;+
                            04C1   1117 ; Destination is intermediate
                            04C1   1118 ;-
           50    04 A0  D0  04C1   1119 80$:    MOVL    DSC$A_POINTER(R0),R0            ; Get destination address
           80    04 AE  7D  04C5   1120         MOVQ    4(SP),(R0)+                     ; Move 8 bytes
           60    0C AE  D0  04C9   1121         MOVL    12(SP),(R0)                     ; Move 4 more bytes
                 50  8E  D0  04CD  1122         MOVL    (SP)+,R0                        ; Restore status
                        04  04D0   1123         RET                                     ; Return
                            04D1   1124 ;
                            04D1   1125         .END
```

```
BAD_DT                            000000F1 R       02          T3                                  = 0000001C
COB$ADDI                          00000114 RG      02          T4                                  = 00000006
COB$CMPI                          00000321 RG      02
COB$CVTDI_R7                      ********    X    00
COB$CVTFI_R7                      ********    X    00
COB$CVTID_R7                      ********    X    00
COB$CVTIF_R7                      ********    X    00
COB$CVTIL_R8                      ********    X    00
COB$CVTIP_R9                      ********    X    00
COB$CVTIQ_R8                      ********    X    00
COB$CVTIW_R8                      ********    X    00
COB$CVTLI_R8                      ********    X    00
COB$CVTPI_R9                      ********    X    00
COB$CVTQI_R8                      ********    X    00
COB$CVTWI_R8                      ********    X    00
COB$DIVI                          0000027E RG      02
COB$DIVI_OSE                      00000277 RG      02
COB$MULI                          00000217 RG      02
COB$SUBI                          000000FE RG      02
COB$_INTDIVZER                    ********    X    00
COB$_INTEXPOVE                    ********    X    00
COB$_INTEXPUND                    ********    X    00
COB$_INTRESOPE                    ********    X    00
COB$_INVARG                       ********    X    00
CONVERT                           00000002 R       02
D1                              = 0000000D
D2                              = 00000006
DD                              = 00000038
DIV_J                             00000282 R       02
DR                              = 0000002C
DSC$A_POINTER                   = 00000004
DSC$B_CLASS                     = 00000003
DSC$B_DTYPE                     = 00000002
DSC$B_SCALE                     = 00000008
DSC$K_CLASS_SD                  = 00000009
DSC$W_LENGTH                    = 00000000
FINISH                            0000039C R       02
INT$K_I_EXP_HI                  = 00000063
INT$K_I_EXP_LO                  = FFFFFF9D
INT$K_I_FRACT_D                 = 00000012
INT$K_I_FRACT_L                 = 0000000A
INT$K_I_LEN                     = 0000000C
INT$P_I_FRACT                   = 00000002
INT$W_I_EXP                     = 00000000
LIB$STOP                          ********    X    00
MD                              = 0000000C
MR                              = 00000000
O1                              = 00000003
OSE                             = 00000044
P0                                00000000 R       02
P1                                00000001 R       02
PR1                             = 00000018
PR2                             = 00000022
SP_AMT                          = 00000045
SP_DECR                         = 0000002F
T1                              = 00000000
T2                              = 00000015
```

```
                                                        +----------------+
                                                        ! Psect synopsis !
                                                        +----------------+

PSECT name                        Allocation          PSECT No.  Attributes
----------                        ----------          ---------  ----------
.  ABS  .                         00000000  (     0.)  00 (  0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                             00000000  (     0.)  01 (  1.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
_COB$CODE                         000004D1  (  1233.)  02 (  2.)    PIC    USR    CON    REL    LCL   SHR   EXE   RD   NOWRT NOVEC LONG

                                                     +------------------------+
                                                     ! Performance indicators !
                                                     +------------------------+

Phase                        Page faults    CPU Time        Elapsed Time
-----                        -----------    --------        ------------
Initialization                        31    00:00:00.06     00:00:02.10
Command processing                   118    00:00:00.37     00:00:03.37
Pass 1                               188    00:00:02.84     00:00:15.02
Symbol table sort                      0    00:00:00.19     00:00:00.47
Pass 2                               196    00:00:01.51     00:00:06.13
Symbol table output                    9    00:00:00.03     00:00:00.05
Psect synopsis output                  3    00:00:00.01     00:00:00.01
Cross-reference output                 0    00:00:00.00     00:00:00.00
Assembler run totals                 547    00:00:05.03     00:00:27.15

The working set limit was 1500 pages.
24129 bytes (48 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 188 non-local and 52 local symbols.
1125 source lines were read in Pass 1, producing 30 object records in Pass 2.
9 pages of virtual memory were used to define 8 macros.

                                                    +--------------------------+
                                                    ! Macro library statistics !
                                                    +--------------------------+

Macro library name                                  Macros defined
------------------                                  --------------
_$255$DUA28:[COBRTL.OBJ]COBRTL.MLB;1                      1
_$255$DUA28:[SYSLIB]STARLET.MLB;2                         4
TOTALS (all libraries)                                    5

203 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS$:COBINTARI/OBJ=OBJ$:COBINTARI MSRC$:COBINTARI/UPDATE=(ENH$:COBINTARI)+LI
```

0063 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY